**Solidity Reverts**

**Introduction:**

In this lesson, we will delve into how do transaction reverts work, what is gas where is used.

**Revert:**

Let’s start by adding some logic to the `fund` function:

uint256 public myValue = 1;

Function fund() public {

myValue = myValue + 2;

}

A revert action undoes all prior operations and returns the remaining gas to the transaction’s sender. In this `fund` function, `myValue` increases by two (2) units with each successful execution. However, if a revert statement is encountered right after, all actions performed from the start of the function are undone. `myValue` will then reset to its initial state value, or one.

Uint256 public myValue = 1;

Function fund() public {

myValue = myValue + 2;

require(msg.value > 1e18, “didn’t send enough ETH”);

// a function revert will undo any actions that have been done.

// It will send the remaining gas back

}

**Gas Usage:**

**Cautions:** The gas used in the transaction will not be refunded if the transaction fails due to a revert statement. The gas has already been consumed because the code was executed by the computers, even though the transaction was ultimately reverted.

Users can specify how much gas they’re willing to allocate for a transaction. In the case where the `fund` function will contain a lot of lines of code after the `require` and we did indeed set a limit, the gas which was previously allocated but not used will not be charged to the user

**Note:** If a transaction reverts, is defined as failed

**Transaction Fields:**

During a value transfer, a transaction will contain the following fields:

* **Nonce:** transaction counter for the account
* **Gas price (wei):** maximum price that the sender is willing to pay per unit of gas
* **Gas Limit:** maximum amount of gas the sender is willing to use for the transaction. A common value could be around 21000.
* **To:** recipient’s address
* Value (Wei): amount of cryptocurrency to be transferred to the recipient
* **Data:** empty
* **v,r,s:** components of the transaction signature. They prove that the transaction is authorised by the sender.

During a contract interaction transaction, it will instead be populated with:

* **Nonce:** transaction counter for the account
* **Gas price (wei):** maximum price that the sender is willing to pay per unit of gas
* **Gas Limit:** maximum amount of gas the sender is willing to use for the transaction. A common value could be around 21000.
* **To:** address the transaction is sent to (e.g. smart contract)
* **Value (Wei):** amount of cryptocurrency to be transferred to the recipient
* **Data:** the content to send to the `To` address, e.g. a function and its parameters.
* **v,r,s:** components of the transaction signature. They prove that the transaction is authorised by the sender.

**Conclusion:**

Reverts and gas usage help maintain the integrity of the blockchain state. Reverts will undo transactions when failures occur, while gas enables transactions execution and runs the EVM. When a transaction fails, the gas consumed is not recoverable. To manage this, Ethereum allows users to set the maximum amount of gas they’re willing to pay for each transaction.